Software Metrics and Problem Detection

Mircea Lungu
Roadmap

> Measurements
> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost
> Metric-Based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems
> Discussion
Measurements

A measurement is a mapping
  domain
  range
  rules

A measure is a numerical value or a symbol assigned during mapping

In Software:
measurements = metrics

Estimation of quantity owes its existence to Measurement Calculation to Estimation of quantity Balancing of chances to Calculation and Victory to Balancing of chances.
Measurement Scales

- Nominal
- Ordinal
- Interval
- Ratio

Analysis should take scales into account.

Estimation of quantity owes its existence to Measurement
Calculation to Estimation of quantity
Balancing of chances to Calculation
and Victory to Balancing of chances.
Medical Markers are used in diagnostics based on statistical data

> Potassium Levels
> Red Blood Cell Count
> Glucose Levels
> etc.
Google Measures N-gram Frequencies

> What do you do when you want to digitize and report about 5 million books but can not because of copyright?
Can you assess unknown code without reading it?
Fraser:
One of the problems that is central to the software production process is to identify the nature of progress and to find some way of measuring it.

McIlroy:
In programming efforts [...] clarity and style seem to count for nothing — the only thing that counts is whether the program works when put in place. It seems to me that it is important that we should impose these types of aesthetic standards.
Roadmap

- Measurements
- **Software Metrics**
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-Based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion
Roadmap

> Measurements

> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost

> Metric-Based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems

> Discussion
Size Measures

LOC
NOM
NOA
NOC
NOP
... etc.

Lorenz, Kidd, 1994
Chidamber, Kemerer, 1994
Cyclomatic Complexity (CYCLO)

The number of independent linear paths through a program.

(McCabe ’77)

+ Measures minimum effort for testing
Weighted Methods per Class (WMC)

The complexity of a class by summing the complexity of its methods, usually using CYCLO.

(Chidamber & Kemerer ’94)

+ A proxy for the time and effort required to maintain a class
The maximum depth level of a class in a hierarchy.  
(Chidamber & Kemerer ’94)

+ Inheritance depth is a good proxy for complexity
Access To Foreign Data (ATFD)

ATFD counts how many attributes from other classes are accessed directly from a given class. (Lanza & Marinescu ’06)

+ ATFD summarizes the interaction of a class with its environment
Roadmap

> Measurements
> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost
> Metric-Based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems
> Discussion
Coupling Between Object Classes (CBO)

CBO for a class is the number of other classes to which it is coupled.  
(Chidamber & Kemerer '94)

+ Meant to assess modular design and reuse
Tight Class Cohesion (TCC)

TCC counts the relative number of method-pairs that access attributes of the class in common.

(Bieman & Kang, 95)

+ Can lead to improvement action

TCC = 2 / 10 = 0.2
Roadmap

> Measurements

> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost

> Metric-Based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems

> Discussion
Man-Month/Year

The amount of work performed by an average developer in a month/year.
FP is a unit of measurement to express the amount of functionality an information system provides to a user.

- Risks hiding the internal functions (algorithms)
The Measurement Process

The Goal-Question-Metric model proposes three steps to finding the correct metrics.

(Victor Basili)

1) Establish the **goals** of your maintenance or development project.

2) Derive, for each goal, **questions** that allow you to verify its accomplishment.

3) Find what should be **measured** in order to quantify the answer to the questions.

Targets without clear goals will not achieve their goals clearly.

Gilb’s Principle
Roadmap

> Measurements

> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost

> Metric-based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems

> Discussion
Roadmap

> Measurements

> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost

> Metric-based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems

> Discussion
The Overview Pyramid provides a metrics overview.

Inheritance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDC</td>
<td>0.31</td>
</tr>
<tr>
<td>AHH</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Size

Communication

Lanza, Marinescu 2006
The Overview Pyramid provides a metrics overview.
The Overview Pyramid provides a metrics overview.

**Communication**
The Overview Pyramid provides a metrics overview.

Inheritance

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDC</td>
<td>0.31</td>
</tr>
<tr>
<td>AHH</td>
<td>0.12</td>
</tr>
</tbody>
</table>
The Overview Pyramid provides a metrics overview.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDC</td>
<td>0.31</td>
</tr>
<tr>
<td>AHH</td>
<td>0.12</td>
</tr>
<tr>
<td>NOP</td>
<td>19</td>
</tr>
<tr>
<td>NOC</td>
<td>384</td>
</tr>
<tr>
<td>NOM</td>
<td>3618</td>
</tr>
<tr>
<td>NOM</td>
<td>418</td>
</tr>
<tr>
<td>LOC</td>
<td>35175</td>
</tr>
<tr>
<td>CALLS</td>
<td>15128</td>
</tr>
<tr>
<td>CYCLO</td>
<td>5579</td>
</tr>
<tr>
<td>FANOUT</td>
<td>8590</td>
</tr>
</tbody>
</table>
The Overview Pyramid provides a metrics overview.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDC</td>
<td>0.31</td>
</tr>
<tr>
<td>AHH</td>
<td>0.12</td>
</tr>
<tr>
<td>NOP</td>
<td>19</td>
</tr>
<tr>
<td>NOM</td>
<td>384</td>
</tr>
<tr>
<td>NOM</td>
<td>418</td>
</tr>
<tr>
<td>LOC</td>
<td>3618</td>
</tr>
<tr>
<td>NOM</td>
<td>418</td>
</tr>
<tr>
<td>CALLS</td>
<td>0.56</td>
</tr>
<tr>
<td>FANOUT</td>
<td>0.56</td>
</tr>
</tbody>
</table>

- close to high
- close to average
- close to low
The Overview Pyramid provides a metrics overview.
How to obtain the thresholds?

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LOW</td>
<td>AVG</td>
</tr>
<tr>
<td>CYCLO/LOC</td>
<td>0.16</td>
<td>0.20</td>
</tr>
<tr>
<td>LOC/NOM</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>NOM/NOC</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Statistical static analysis of reference systems
Context is important (e.g. programming language)
Roadmap

- Measurements
- Software Metrics
  - Size / Complexity Metrics
  - Quality Metrics
  - Schedule / Cost
- Metric-based Problem Detection
  - Detecting Outliers
  - Encoding Design Problems
- Discussion
Design Problems and Principles

Bad Smells
- Comments
- Switch Statement
- Shotgun Surgery
...

Design Heuristics
- Encapsulation
- Minimize Coupling
- Class Coherence
- Inheritance Depth
...

Design principles come in prose - how to measure them?
Rarely a single metric is sufficient >>> Detection Strategies
Detection Strategies...

... are metric based queries for detecting design problems (Lanza & Marinescu 2002)
God Classes ...

... tend to **centralize the intelligence** of the system, to do everything, and to **use data** from small data-classes
God Classes ...

... tend to **centralize the intelligence** of the system, to **do everything**, and to **use data** from small data-classes

- **Complexity (WMC)**

- **Lack of cohesion (TCC)**

- **Foreign data usage (ATFD)**
God Classes

- Class uses directly more than a few attributes of other classes
  - ATFD > FEW
- Functional complexity of the class is very high
  - WMC ≥ VERY HIGH
- Class cohesion is low
  - TCC < ONE THIRD

Quantifiers
- FEW
- MANY
- TOP
- HIGH
- ONE THIRD ...
Data Classes are dumb data holders

![Diagram showing the relationship between interface of class, WOC, AND, and Data Class.]

<table>
<thead>
<tr>
<th>WOC - Weight Of a Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>The number of “functional” public methods divided by the total number of public members (Mar02a)</td>
</tr>
</tbody>
</table>
Data Classes are dumb data holders

NOAP = #Public Attributes,
NOAM = #Accessor Methods
Feature Envy is ...

This one you find in the Lanza-Marinescu Book!
Roadmap

> Measurements
> Software Metrics
  — Size / Complexity Metrics
  — Quality Metrics
  — Schedule / Cost
> Metric-based Problem Detection
  — Detecting Outliers
  — Encoding Design Problems
> Discussion
McClure:
I know of one organisation that attempts to apply time and motion standards to the output of programmers. They judge a programmer by the amount of code he produces. This is guaranteed to produce insipid code — code which does the right thing but which is twice as long as necessary.
FAMIX 3.0

- Meta-model
- Core - independent of programming language
- Implemented in Moose
NamedEntity
- isFinal: Boolean
- isProtected: Boolean
- name: String
- isPackage: Boolean
- isAbstract: Boolean
- isPrivate: Boolean
- isPublic: Boolean
- belongsTo: ContainerEntity
- modifiers: String*
- isStub: Boolean
- parentPackage: Package -> childNamedEntities
- receivingInvocations: Invocation* -> receiver

Entity

SourcedEntity
- sourceAnchor: SourceAnchor -> element
- comments: Comment* -> container

Comment
- content: String

Association
- next: Association -> previous
- previous: Association -> next
- from: NamedEntity
- to: NamedEntity

Inf
- subclass: Type
- superclass: Type
What you should know!

- Software metrics are measurements
- Every scale allows certain operations and analyses
- Detection strategies are queries for design problem detection
- The Goal Question Metric model has three phases
- Bad smells encode bad OO practices
- Design heuristics encode good OO practices
Can you answer these questions?

- How do you compute TCC for a given class?
- Can you explain how the God Class detection strategy works?
- Can you list several of the elements of the FAMIX metamodel?
- What are three metrics appropriate for OO systems but not be appropriate for procedural systems?
- Can you give examples of three bad smells?
- Why are comments a bad smell? But switch clauses?
- Can you give examples of three design heuristics?
Further Reading

> *Cohesion and Reuse in Object Oriented Systems*, by Bieman & Kang
> *OOMIP* by Lanza and Marinescu (Sections 5.3 - 5.5)
> [http://scg.unibe.ch/staff/mircea/sde/60-design-heuristics](http://scg.unibe.ch/staff/mircea/sde/60-design-heuristics)
Attribution-ShareAlike 3.0

You are free:
- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions:

**BY:** Attribution. You must attribute the work in the manner specified by the author or licensor.

**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

http://creativecommons.org/licenses/by-sa/3.0/